中文多模式匹配算法性能的分析与研究

朱永强[1,2],江 雪[1,2]
(1. 成都网安科技发展有限公司,四川 成都 610092;
2. 电子科技大学 示范性软件学院,四川 成都 610054)

摘 要:模式匹配算法一般不具有所有环境下的通用性,不同的算法在不同语义环境下的表现,往往差异较大。为实现中文环境下对模式串的快速多模式匹配,选择出在中文环境下的最优匹配算法,分析了几种经典的多模式匹配算法。通过对各个算法设计思路、时间性能与空间性能的研究,推导出基于"坏字符"的算法设计思路最适用于中文环境下大字符集、短字符串的特点,并通过实验对理论推测的中文环境最优算法-Wang 算法的性能与其他几种经典算法的性能进行了比较,验证了理论推导的正确性。

关键词:多模式匹配;中文环境;AC 算法;WM 算法;Wang 算法
中图分类号:TP301.6 **文献标识码**:A **文章编号**:1673-629X(2014)02-0067-04
doi:10.3969/j.issn.1673-629X.2014.02.016

Analysis and Research of Chinese Multi-pattern Matching Algorithm Performance

ZHU Yong-qiang[1,2], JIANG Xue[1,2]
(1. Chengdu Wang'an Keji Co., Ltd, Chengdu 610092, China;
2. School of Software, UESTC, Chengdu 610054, China)

Abstract: Generally, pattern matching algorithms do not have the versatility of all circumstances. For realizing the fast multi-pattern matching, selecting the optimal matching algorithm under the Chinese environment, analyze several common multi-pattern matching algorithm. By researching the various algorithm design ideas, the time and space performance, deduced that the design idea based on the "bad character" is the best way which can be used to fast matching under Chinese environment, and the experiment shows that the Wang algorithm is the optimal algorithm under Chinese environment compared with other classical algorithm, and verifies the correctness of theory deduction.

Key words: multi-pattern matching; Chinese environment; AC algorithm; WM algorithm; Wang algorithm

0 引 言

模式匹配算法按照模式串数量的不同,可分为单模式匹配与多模式匹配。多模式匹配在实际中有着更广泛的应用,其算法的性能往往与编码环境有关,对于特定的编码环境,不同算法的性能会有较大差别,因此需特定分析使用环境下的特点,选取最优算法,才能使实际应用中获得更高的时间性能。

文中在中文匹配环境下,分析了三种经典多模式匹配算法:AC 算法、WM 算法与 Wang 算法的特点与思路,并在理论分析的基础上,对各算法在中文环境下的性能进行了测试,分析比较出中文检索环境下的最优算法。

1 几种经典的多模式匹配算法简介

1.1 Aho-Corasick 算法

Aho-Corasick 算法[1](简称 AC 算法),于 1975 年由 A. V. Aho 和 M. J. Corasick 提出。AC 算法是多模式匹配中的经典算法,它利用有限自动状态机,将多模式匹配过程中字符的比较操作转换为自动状态机中的状态切换,因此对匹配串进行一次扫描即完成对所有模式串的匹配。AC 算法主要通过转向函数 Goto、失效函数 Failed 与输出函数 Output 来完成多模式匹配。下面

收稿日期:2013-05-02 **修回日期**:2013-08-04 **网络出版时间**:2013-11-29
基金项目:四川省科技型中小企业创新基金(10C26215122841)
作者简介:朱永强(1987-),男,硕士研究生,CCF 会员,研究方向为算法设计。
网络出版地址:http://www.cnki.net/kcms/detail/61.1450.TP.20131129.0857.014.html

简单介绍这三个辅助函数：

转向函数(Goto 函数)：此函数用于确定当前输入自动机的字符 c 在状态机中下一步转向方向，即：Goto(now,c)= next，如果在自动状态机中不存在这样的转换，亦即自动状态机在此输入下无有效状态，则 next = FailState，称 FailState 节点为当前失效点的失效跳转点，其具体节点地址由 Failed 函数确定。Goto 函数在使用前通过依次将模式串输入自动状态机进行初始化，对模式串集合{arda,apple,care}构造自动状态机，生成的 Goto 函数如图 1 所示。

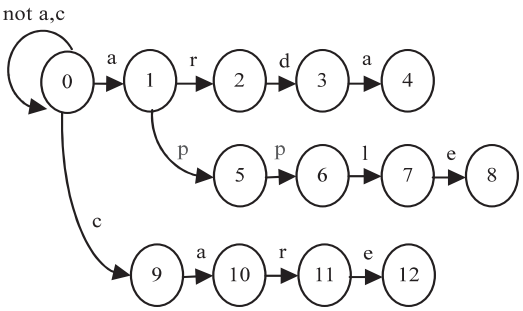

图 1 AC 算法自动状态机

失效函数(Failed 函数)：Failed (now) = Failednext，标识着在 now 节点发生转向失效的情况下(即 Goto 函数不存在 next 节点)，自动状态机的下一个转换状态 Failednext。Failednext 状态节点的特征是：从此状态节点向上直到根节点(状态 0)所经历的所有输入字符路径，与从产生失效状态的节点 now 向上所经历的输入字符串或其子串完全相同。如果此类节点有多个，则选择这些节点中深度最大的节点。如果不存在满足条件的状态节点，则失效函数指向 0 状态节点。

输出函数(Output 函数)：用于当某个模式串匹配完成后，输出当前的匹配信息以及其他的一些标识信息，需要注意的是，如果模式串集合中存在着包含与被包含关系，则应在到达输出状态时同时输出当前的匹配信息与当前匹配信息包含的模式串信息，防止匹配输出漏解。

1.2 Wu-Manber 算法

Wu-Manber 算法[2-3](简称 WM 算法)，是 Sun Wu 和 Udi Manber 于 1994 年提出的一种多模式匹配算法。WM 算法一定程度上借鉴了单模式匹配算法—BM 算法[4-5]的好后缀思想，通过构造三张辅助表：SHIFT、HASH 和 PREFIX 来实现快速准确的匹配，三个表的作用分别如下：

SHIFT[h]表：参数 h 为当前窗口的最后 Length 长度字符串计算所得的哈希值，在扫描文本串的时候，根据当前读入窗口的匹配串末尾 Length 个字符的哈希值 h，查找 SHIFT[h]表，以确定可以安全跳过的字符数，如果跳跃值为 0，则可能产生匹配，需用 HASH 表和 PREFIX 表进一步判断。

HASH[h]表：表示模式串中最后 Length 个字符的哈希值为 h 的所有模式串的链表，是当 SHIFT[h]值为 0 时的所有可能发生匹配的模式串的集合。

PREFIX 表：前缀表，其大小等于模式串个数，由各个模式串的一定长度的前缀字符的哈希值生成，用于进一步过滤当前可能发生匹配的窗口下具有相同后缀 HASH 值的不同模式串，以缩小可能在窗口内发生匹配的模式串范围，减少实际比较的次数。

WM 算法的匹配过程可简单概括如下：

(1)计算匹配串在当前扫描窗口中的最后 Length 个字符的哈希值 h，并查找 SHIFT[h]表。

(2)如果 SHIFT[h]>0，则向后跳跃 SHIFT[h]内的距离，并转第 1 步继续；如果 SHIFT[h] = 0，则说明窗口内可能发生匹配，进行第 3 步。

(3)计算当前匹配窗口的前缀的哈希值，并根据 PREFIX 表，排除一部分后缀局部匹配，而前缀不匹配的情况，以减小模式串局部匹配的影响。

(4)对于 HASH[h]指向列表的每个字符串，依次从后向前匹配当前窗口内字符，检查是否存在匹配。

(5)将当前窗口向后移动一个字符，转向步骤 1 继续进行匹配，直到扫描完全部文件。

1.3 Wang 算法

王永成在 AC 算法的基础上，通过构建 Goto 函数以及 Skip 函数，实现了一种新的多模式匹配算法[6-8]。此算法的核心思想类似于单模式匹配中的 Sunday 算法[9]，算法充分利用了匹配过程中本次匹配不成功的信息，采用“坏字符”的启发式判断规则，尝试跳过最大距离的安全字符，以增大整体平均步进的期望值，提升算法的匹配执行速度。Wang 算法主要使用两个函数(状态转移函数 Goto 和辅助跳转函数 Skip)进行匹配，以下分别介绍这两个函数：

对于 Goto 函数的构造，其基本的生成方法与 AC 算法的自动状态机类似，为 AC 算法自动状态机的逆向反构形式，即使用模式串组中的各个模式串，以从后向前的输入方式，对状态机进行初始化。图 2 给出了同样使用{arda,apple,care}构造的逆向自动状态机。

Skip(char)函数是 Wang 算法的跳转函数，它直接体现了 Wang 算法的“坏字符思想”，通过考察当前与文本串 TXT 对齐的匹配窗口的最右边界字符 i 的下一个字符 char，确定可以跳转的距离，其数学形式为 Skip(TXT[i+1])，这个值表示在输入 char 字符的情况下，算法所能进行的最大安全跳转距离，具体的值，等于字符 char 在各个模式串中最右出现的位置到该模式串串尾的距离 distance 加 1。由于 Skip(char)函数考查当前匹配窗口最右字符的下一个字符，因此算法可获

得的最大跳转距离为 minlen+1,其中 minlen 为当前模式串组中的最小长度。超过此值,则可能漏过模式串集合中最短模式串。

Skip(char)函数的跳转规则可由下式简单概括:

$$\text{Skip(char)} = \begin{cases} \min\{s+1 \mid p_i[m_i - s] = \text{char}, 0 \leqslant s < \text{minlen} \\ \quad \text{且 } 1 \leqslant i \leqslant z\}, \text{char 出现在模式 } P \text{ 中} \\ \text{minlen}+1, \text{char 不出现在模式 } P \text{ 中} \end{cases}$$

其中,s 为当前失配字符距当前模式串最左端的距离,z 为匹配串长度。

Wang 算法从文本串起始点的 minlen 位开始匹配,算法的匹配窗口从左向右移动。设某一次匹配从 TXT[i–j]开始,j 的初始值为 0,按照 Goto 函数进行自右向左的匹配,如果匹配成功,则令 j=j+1,匹配下一个 TXT[i–j];如果匹配失败,则通过 Skip 函数读取此时的跳转值,计算下一个比较点的位置,即 i=i+Skip(TXT[i+1]),然后从此点重新开始一次匹配,若某个字符在反向自动状态机中可以走到终端节点(如图 2 中的 4 号节点、9 号节点),则表示发生一次匹配,可随即进行相应的匹配处理,处理结束后,读取当前窗口的下一个字符进入状态机,继续进行匹配直至结束。

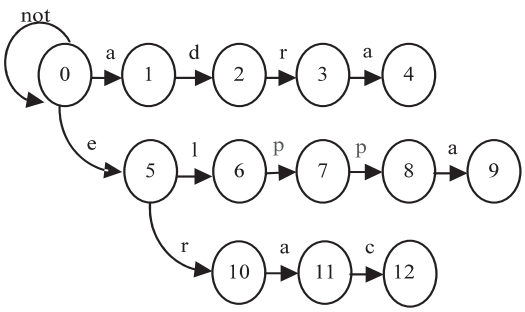

图 2　Wang 算法的反向自动状态机

2　中文环境下的最优匹配算法分析

中文检索的语义环境简单地概括为:大字符集、短模式串[10]。所谓大字符集,指的是中文编码字符的取值范围较大,如使用 GB2321 编码方式,则每个字符的编码范围为 0 至 255,如使用 Unicode 编码方式,则每个中文字符变量的取值范围可达到 0 至 65 535,相对于英文字符最常用的 26 个字符,其字符范围非常大;所谓短模式串,是指中文语义下的关键字平均长度较短,普遍为二或三,其关键字串长度也一般小于英文关键字串。

对于编码环境,中文普遍使用 GB2312 编码或 Unicode 编码,由于算法存储空间的特性,实际处理 Unicode 编码时也往往将 Unicode 码值理解为两个单字节码值的组合,因此在实际对中文关键字进行处理时,虽然每个中文都使用两个字节为一个整体,但往往将每个字节理解为两个具有更小编码范围的单字节进行处理。

以下分别分析此环境下现有算法的空间与时间性能。

2.1　多模式匹配算法的空间性能分析

AC 算法与 Wang 算法使用自动状态机来实现多模式匹配。GB2312 编码环境下,考虑空间存储的问题,若将每个汉字理解为两个单字节的组合,则每个自动状态机标示下一个状态的结构体,需存放 256 个下一节点的地址,若每个地址使用 32 位指针存储,则每个节点需占用:

4×256 = 1 024 byte

设模式串组包含 m 个模式串,平均长度为 n,则消耗空间为:

$$1\,024 \times 2 \times m \times n = 2\,048mn \text{ byte}$$

Wang 算法除了使用自动状态机,还使用了 Skip 函数,Skip 函数的大小则取决于码值的编码范围,题设环境中,同为:

$$4 \times 256 = 1\,024 \text{ byte}$$

由此可知,AC 算法的空间消耗为 2 048mn byte,而 Wang 算法为 1 024(2mn+1) byte。

WM 算法存储三个辅助表,其中 PREFIX 表的大小取决于模式串组中模式串的数目,一般来说,可忽略不计,而 HASH[h]表与 SHIFT[h]表的大小取决于其窗体内 HASH 值的取值范围,设 HASH 的范围值为 q,存储 32 位跳转值,则

$$4 \times 2 \times q = 8q \text{ byte}$$

一般情况下,q 值小于当前的编码范围,即 8q 小于 2 048 byte。

综上,在空间消耗上,AC 算法与 Wang 算法基本相同,约为 WM 算法空间消耗的 mn 倍。而三种算法的空间消耗数量级为 K 字节,对于当代计算机的性能,几乎可以忽略不计,因此下面主要分析各个算法的时间性能。

2.2　多模式匹配算法的时间性能分析

设中文使用 GB2312 方式进行编码,文本串的长度为 n,模式串集合中最短长度为 m,算法运行于最优环境下(即算法达到理论最大性能的环境),以下分析各个算法在当前编码的中文环境下的性能。

AC 算法依靠输入字符在自动状态机中的状态转换进行匹配,其每次比较后的步进恒为 1,AC 算法没有利用中文字符特点,利用坏字符进行跳跃,因此性能较差,其时间消耗恒定为 $O(n)$。

WM 方法通过计算窗口内后缀的哈希散列,将后缀的多个字符整合为一个哈希值,以减少匹配的实际比较次数,通过前缀表,减少后缀匹配的前提下可能发

生的部分匹配,并利用 SHIFT[h]表进行向后跳转。SHIFT[h]表的跳转值大小,受其最小模式串长度的影响,实际上,WM 算法所能达到的最大跳转距离便为其最短模式串长度 minlen。如果最短模式长度过短,则 SHIFT[h]表的最大位移也将受到影响。另一方面,当模式串过短时,WM 算法中的 HASH[h]表与 PREFIX 表之间的相隔码距也会变小,甚至会有部分重合,这将导致 PREFIX 表对模式串部分匹配情况的过滤作用下降。因此可以推测,一旦模式串过短,WM 算法的效率会受到较大影响。

此外,对于每个窗口内字符的哈希值计算,也将消耗一定的运算时间。

最优环境下,WM 算法的时间效率可以达到 $O(n/m)$,由于哈希值的计算以及前缀表的比较,真实的最优时间要略高于 $O(n/m)$。

Wang 算法利用自动状态机进行状态切换;利用匹配中的失配字符,通过 Skip 函数进行跳转,Skip 函数的最大跳转值可以取到 minlen+1。由于中文语义环境下的字符集较大,字符值散度较大,一般情况下,匹配串中出现模式串字符的几率非常小,因此 Skip 函数取得最大跳转值的可能性也较大(即所有模式串中都不包括考察字符)。由于充分利用了坏字符特性,使得 Wang 算法理论上在中文语义环境下,相较于其他算法有着更高的时间效率。最优环境下,Wang 算法的时间效率可以达到 $O[n/(m+1)]$。

可以看到,在中文语义下,由于字符散度较大,WM 算法与 Wang 算法取得最大跳转值的几率都比较大,而 AC 算法的跳转值则恒定为 1。

综上可知,Wang 算法通过每次比较获取的跳转值期望要高于 AC 算法与 WM 算法(分别为 minlen+1、minlen、1)。而其坏字符的设计思路也符合中文检索环境的大字符集特点,并且不需要额外的附加运算(如计算哈希值),因此理论上,Wang 算法为中文检索环境下的最高效算法。

2.3 各种算法的具体执行流程

下面以实际应用流程为例,进一步分析各个算法在匹配执行时的时间性能:

设模式串组 PAT:{四川,成都},匹配串 TXT:{纵死侠骨香,不惭世上英},字符显示使用 16 进制,每个算法分别对 TXT 匹配三次,观察三次匹配后三种算法的移动距离,则模式串组编码如下:

PAT:{cb c4 b4 a8,b3 c9 b6 bc}

三种算法对于 TXT 的匹配流程见表 1。

可以看到,三次比较之后,AC 算法移动到 TXT 的第 3 个字符,WM 算法移动到第 12 个字符,Wang 算法可以移动到 14 个字符,具有最高效的时间性能,实际流程与理论推测相符。

表 1 三种算法的匹配过程

比较轮数	比对字符串位置
第一次比较字符	AC WM、Wang ↓ ↓ d7 dd cb c0 cf c0 b9 c7 cf e3 a3 ac b2 bb b2 d1 ca c0 c9 cf d3 a2
第二次比较字符	AC WM Wang ↓ ↓↓ d7 dd cb c0 cf c0 b9 c7 cf e3 a3 ac b2 bb b2 d1 ca c0 c9 cf d3 a2
第三次比较字符	AC WM Wang ↓ ↓ ↓ d7 dd cb c0 cf c0 b9 c7 cf e3 a3 ac b2 bb b2 d1 ca c0 c9 cf d3 a2

3 实验结果与分析

对算法进行实验测试,获取时间性能的相关数据。样本选取人民日报 2009 至 2011 年的社论摘要,共计 83.566 3 万字。模式串组为在字典中随机抽取的四组,每组包含 7 个模式串,最短模式串长度从 2 至 5。实验只记录总的匹配数,且只考察算法进行匹配的执行时间。实验采取 16 进制显示,GB2312 编码,实验数据表格与实验结果曲线分别如表 2 和图 3 所示。

表 2 最短模式串长度不同时的实验数据表 Counts

最短模式串长度	2	3	4	5
AC 算法	19 963	20 135	21 967	22 003
WM 算法	18 999	12 265	8 251	6 410
Wang 算法	11 782	9 647	7 652	5 980

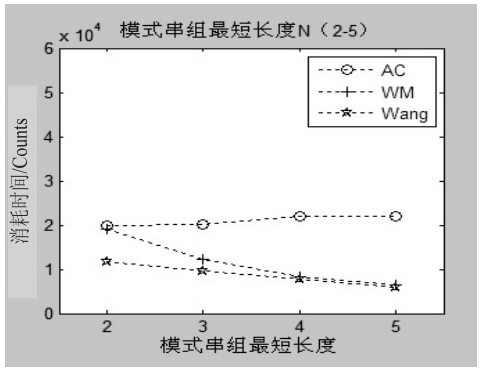

图 3 最短模式串长度取不同值时的各算法时间性能消耗

根据实验数据,WM 算法与 Wang 算法的效率都会随着最短模式长度降低而下降,这是由于这两种算法的最大跳转距离都依赖于最短模式串长度。

各实验环境下,AC 算法的时间效率都是最低的,因此在中文环境下,虽然 AC 算法对最短模式不敏感,但其时间性能最差,并不适用于中文信息匹配与检索。

Wang 算法在各个测试情况下,其时间效率均是同等环境下最高效的,这证明了 Wang 算法在中文检索环境下的优势与稳定性,但是随着最短模式串m的增

(下转第 75 页)

6　结束语

网格计算环境中的工作流调度问题是目前研究的一大热点,针对现有的调度算法的不足,文中提出了一种改进的调度算法。该算法在考虑计算站点负载和调度任务的截止时间约束的前提下,采用一种启发式的方法实现从任务资源需求到计算站点的映射。仿真实验结果表明,文中的方法是有效的,优于两种实际网格中的调度算法。下一步研究工作的重点在于:

1)综合考虑资源的可利用性和可靠性来设计工作流的快速调度算法;

2)考虑实际应用环境下的工作流调度问题建模。

参考文献:

[1] 李　玺,胡志刚,胡周君,等. 基于截止时间满意度的网格工作流调度算法[J]. 计算机研究与发展,2011,48(5):877-884.

[2] 王　勇,胡春明,杜宗霞. 服务质量感知的网格工作流调度[J]. 软件学报,2006,17(11):2341-2351.

[3] Topcuoglu H, Hariri S, Wu M. Performance effective and low-complexity task scheduling for heterogeneous computing[J]. IEEE transactions on parallel and distributed systems, 2002, 13(3):260-274.

[4] Hunold S, Rauber T, Suter F. Scheduling dynamic workflows onto clusters of clusters using postponing[C]//Proceedings of the 8th IEEE international symposium on cluster computing and the grid. Lyon, France: IEEE Computer Society, 2008:669-674.

[5] 苑迎春,李小平,王　茜,等. 成本约束的网格工作流时间优化方法[J]. 计算机研究与发展,2009,46(2):194-201.

[6] 肖　鹏,胡志刚. 截止时间约束下独立网格任务的协同调度模型[J]. 电子学报,2011,39(8):1852-1857.

[7] Wieczorek M, Podlipnig S, Prodan R, et al. Bi-criteria scheduling of scientific workflows for the grid[C]//Proceedings of the 8th IEEE international symposium on cluster computing and the grid. Lyon, France: IEEE Computer Society, 2008:9-16.

[8] 苑迎春,李小平,王　茜,等. 基于优先级规则的网格工作流调度[J]. 电子学报,2009,37(7):1457-1464.

[9] Berten V, Goossens J, Jcannot E. On the distribution of sequential jobs in random brokering for heterogeneous computational grids[J]. IEEE trans on parallel and distributed systems, 2006, 17(2):113-124.

[10] Gross D, Harris C M. Fundamentals of queuing theory[M]. New York: John Wiley and Sons, 1998.

[11] Josup A, Jan M, Sonmez O O, et al. The characteristics and the performance of groups of jobs in grids[J]. Lecture notes on computer science, 2007, 46(44):382-393.

[12] 刘刚国,罗省贤. 基于指标体系的网格调度算法研究与实现[J]. 计算机工程与应用,2012,48(15):97-101.

(上接第 70 页)

加,Wang 算法与 WM 算法的理论速度比:$q = (m + 1)/m$ 将递减,即两种算法随着最短模式串的增加,呈现逼近的趋势,因此实验数据与理论测试相符。

由于中文语义下的模式串平均长度较短,因此在一般情况下,Wang 算法的时间性能是要明显优于 WM 算法的,是中文环境下多模式匹配算法的首选算法。

4　结束语

文中结合实际的中文语义检索环境,分析了中文环境的语义与编码特点,对三种代表不同思路的多模式算法进行了此环境下的理论分析与实验测试,得到了中文环境下的最优多模式匹配算法,并通过实验对推测加以证明。可以看到,在中文检索与匹配的实际应用中[11-12](如:电子文档检索、网络数据包关键信息过滤),使用 Wang 算法将会获得最优的时间性能。

参考文献:

[1] Aho A V, Corasick M J. Efficient string matching: An aid to bibliographic search[J]. Communication of the ACM, 1975, 18(6):333-340.

[2] Wu Sun, Manber U. Agrep-A fast approximate pattern-matching tool[C]//Proc of USENIX winter technical conference. [s. l.]: [s. n.], 1992:153-162.

[3] Wu S, Manber U. A fast algorithm for multi-pattern searching[R]. Arizona: University of Arizona at Tuscon, 1994.

[4] 张红梅,范明钰. 模式匹配 BM 算法改进[J]. 计算机应用研究,2009,26(9):3249-3252.

[5] Boyer R S, Moore J S. A fast string searching algorithm[J]. Communications of the ACM, 1977, 20(10):762-772.

[6] 王永成,沈　州,许一震. 改进的多模式匹配算法[J]. 计算机研究与发展,2002,39(1):55-60.

[7] 李伟男,鄂跃鹏,葛敬国,等. 多模式匹配算法及硬件实现[J]. 软件学报,2006,17(12):2403-2415.

[8] 章　张. 基于层次分类的网络内容监管系统中串匹配算法的设计与实现[D]. 南京:南京理工大学,2004.

[9] Sunday D M. A very fast substring search arithmetic[J]. Communications of the ACM, 1990, 33(8):132-142.

[10] 孙钦东,黄新波,王　倩. 面向中英文混合环境的多模式匹配算法[J]. 软件学报,2008,19(3):674-686.

[11] 尹中航,王永成,蔡　巍,等. 利用串匹配技术实现网上新闻的主题提取[J]. 软件学报,2002,13(2):159-167.

[12] 高朝勤,陈元琰,李　梅. 一种面向入侵检测的快速多模式匹配算法[J]. 计算机应用,2008,28(1):82-84.

中文多模式匹配算法性能的分析与研究

作者： 朱永强， 江雪， ZHU Yong-qiang， JIANG Xue

作者单位： 成都网安科技发展有限公司，四川 成都 610092；电子科技大学 示范性软件学院，四川 成都610054

刊名： 计算机技术与发展

ISTIC

英文刊名： Computer Technology and Development

年，卷(期)： 2014(2)